1/5/1
DIALOG(R)File 347:JAPIO

00850457 **Image available**
JOB EXECUTION SCHEDULE SYSTEM

PUB. NO.: 57-000757 A]
PUBLISHED: January 05, 1982 (19820105)
INVENTOR(s): YAMAJI HIDEKAZU
ONO MIE
APPLICANT(s): HITACHI LTD [000510] (A Japanese Company or Corporation), JP (Japan)
APPL. NO.: 55-074258 [JP 8074258]
FILED: June 04, 1980 (19800604)
INTL CLASS: [3] G06F-009/46; G06F-015/16
JAPIO CLASS: 45.1 (INFORMATION PROCESSING -- Arithmetic Sequence Units); 45.4 (INFORMATION PROCESSING -- Computer Applications)
JOURNAL: Section: P, Section No. 110, Vol. 06, No. 59, Pg. 100, April 16, 1982 (19820416)

ABSTRACT

PURPOSE: To control dynamically the number of execution jobs so that the processing performance of each processor is maximized, by controlling a load on the processor by periodically grasping an actual load on the processor.

CONSTITUTION: Loads on processors 1 and 2 are measured periodically by a load measuring mechanism 50, and an adequate load deciding mechanism 49 provided to the global processor 1 decides on the proprieties of job execution of the processors 1 and 2 on the basis of sent load information. According to this proper multiplicity, fetching jobs enqueued to be executed is controlled to make the loads on the processors 1 and 2 proper.

⑲ 日本国特許庁（JP）　⑪ 特許出願公開

⑫ 公開特許公報（A）　昭57—757

⑤Int. Cl.³	識別記号	庁内整理番号	⑬公開　昭和57年(1982)1月5日
G 06 F 9/46		6745—5B	
15/16		7165—5B	

発明の数　1
審査請求　未請求

（全 9 頁）

⑭ジョブ実行スケジュール方式

⑳特　　願　昭55—74258
㉒出　　願　昭55(1980)6月4日

⑫発 明 者　山路英一
国分寺市東恋ケ窪1丁目280番地株式会社日立製作所中央研究所内

⑫発 明 者　大野美恵
国分寺市東恋ケ窪1丁目280番地株式会社日立製作所中央研究所内

⑪出 願 人　株式会社日立製作所
東京都千代田区丸の内1丁目5番1号

⑭代 理 人　弁理士　薄田利幸

明　細　書

発明の名称　ジョブ実行スケジュール方式

特許請求の範囲

複数のジョブを処理するプロセサを1台以上有する電子計算機システムにおいて、プロセサにかかっている負荷を定期的に計測し、計測された負荷情報をもとにプロセサのジョブ実行の適正多重度を決定し、決定された適正多重度に従うようにジョブの実行をスケジュールすることを特徴とするジョブ実行スケジュール方式。

発明の詳細な説明

本発明は、電子計算機システムのジョブの実行スケジュールの方式に関するものである。

1台のプロセサ（中央処理装置）から構成されるシングル・プロセサ・システムの処理能力面でのあい路を解消するため、または、信頼性向上を図るために、今後、複数台のプロセサを結合したマルチ・プロセサ・システムが増えていくことが予想される。マルチ・プロセサ・システムは、それを構成する各プロセサの結合の仕方により、いくつかの種類に分けられるが、その1つに、各プロセサでＤＡＳＤ（Direct Access Storage Device：直接アクセス記憶装置（ディスク装置等）を共有し、ＣＴＣＡ（Channel To Channel Adaptor：チャネル間結合装置）を用いて、分散型に結合した複数システムがある。

これを、各プロセッサが主記憶装置を共有して1つのＯＳ（オペレーティング・システム）の下で動くＴＣＭＰ（Tightly Coupled Multi Processor：密結合マルチ・プロセサ）システムに対して、ＬＣＭＰ（Loosely Coupled Multi processor：疎結合マルチ・プロセサ）システムと呼ぶ。

第1図に3台のプロセサからなるＬＣＭＰシステムの構成例を示す。この場合、1台のプロセサ（グローバル・プロセサと呼ぶ）1がカード・リーダ3等からのジョブの入力ならびにラインプリンタ4等へのジョブ実行結果の出力を行なうとともに、自プロセサ1ならびに他プロセサ（ローカル・プロセサと呼び）2へのジョブの分配を行な

う。グローバル・プロセサ1と各ローカル・プロセサ2は、ジョブに関する各種のデータの受け渡しのため、チャネル・スイッチ（図示せず）を介してDASD上のスプール・ボリューム5を共有する他に、ジョブの取り出し要求、ジョブ実行の終了通知等の制御情報の受け渡しのためにCTCA6を介して接続されている。ジョブの入出力はグローバル・プロセサ1で行なうが、TSS（Time Sharing System）端末7はローカル・プロセサ2にも接続可能であり、TSSジョブはTSS端末が接続されているプロセサで実行される。

第2図にLCMPシステムにおけるジョブ処理の一般的な構成を示す。ユーザは各自のジョブをグローバル・プロセサ1に接続されたカード・リーダ3等から入力リーダ24により読み込ませ入力する。入力されたジョブは実行待ちジョブ・キュー25を形成する。実行待ちの間、ジョブはスプール・ボリューム5（第1図）内に格納されている。ジョブ・スケジューラ26はジョブの実行をスケジュールするもので、各プロセサ（グローバル・プロセサ1およびローカル・プロセサ2）のジョブ実行を司るイニシエータ27からジョブの取り出し要求があると、実行待ちジョブ・キュー25の中から、通常は、先に入力された順にジョブを取り出し、イニシエータに渡す。イニシエータ27の制御によりジョブの実行が行なわれた後、ジョブ実行の終了通知がジョブ・スケジューラ26になされる。このとき、ジョブの実行結果はスプール・ボリューム5に格納されており、実行後のジョブは出力待ちジョブ・キュー28を形成する。そしてジョブの実行結果は出力ライタ29により逐一スプール・ボリューム5からラインプリンタ30等に出力される。

一般に、マルチ・プロセサ・システムを構成し十分な性能を引き出すためには、システムにかかる負荷を各プロセサにバランスよく分配し、各々のプロセサにその能力を最大限に発揮させることが必要である。然るに、上記で述べたように従来のシステムにおいては、各プロセサのイニシエータ27からジョブの取り出し要求があると、当該プロセサにおけるTSSジョブを含めた実負荷とは無関係に、予め指定されたイニシエータ27の数に応じてジョブがスケジュールされる。従って、各プロセサの負荷の変動（例えば、アクティブなTSS端末の増減）に対処するため、オペレータがその都度イニシエータの数を変更するなどしてプロセサにかかる負荷を適切化しなければならない。しかし、イニシエータ数が不適切であるとオペレータが知るのは、プロセサの処理能力が大巾に低下した時点になってからであり、時々刻々適切なイニシエータ数の制御ができない。このため、処理能力を最大になるように制御することが困難である。

本発明の目的は、プロセサの処理能力を最大限にするように、実行ジョブ数を動的に制御可能にすることにある。

本発明の特徴とするところは、各プロセサにかかっている実負荷を定期的に把握して、プロセサにかかる負荷を動的に制御することにある。

以下、本発明を実施例によって詳細に説明する。

第3図はグローバル・プロセサ1とローカル・プロセサ2をジョブの実行スケジュールに関する処理部分のみを取り上げて表わしてある。この図で表示されている各機構は、本発明を実行するための制御プログラムを機能別に分けて表示したものである。

負荷計測機構50は各プロセサごとに設けられ、一定時間間隔（計測インタバル）ごとにプロセサの負荷を計測し、グローバル・プロセサ1の適正負荷決定機構49へ負荷情報として送信する。負荷情報としてジョブの平均処理速度係数ASRとスループット係数TRを計測する。あるインタバルにおけるジョブiの処理速度係数SRiジョブ平均処理速度係数ASR、スループット係数TRは次の式で定義されるものである。

ジョブiの処理速度係数SR_i

$$= \frac{CPU_i + IO_i}{ETIME_i} \quad \cdots\cdots\cdots (1)$$

ここに

CPU_i：そのインタバルにおけるジョブiのCPU（中央処理装置）処理時間

IO_i：そのインタバルにおけるジョブiの入出力処理時間

$ETIME_i$：そのインタバルにおけるジョブiの経過時間（経過時間の始点、終点は、当該ジョブが当該インタバルの途中で開始／終了していない限りインタバルの始点、終点に等しく、経過時間はインタバルの長さに等しい。当該ジョブが当該インタバルの途中で開始したときは、経過時間始点はこの開始時刻に等しく、当該ジョブが当該インタバルの途中で終了したときは、経過時間の終点は、この終了時刻に等しく選ばれる。但し、ジョブが端末からの入力待ち等のために"長時間待ち状態"になった場合は経過時間に入れない）

ジョブの平均処理速度係数ASR

$$= \sum_{i=1}^{N} \frac{ETIME_i}{\sum_{i=1}^{N} ETIME_i} \times SR_i \quad \cdots\cdots\cdots (2)$$

ここに　N：当該インタバルにおける実行ジョブ数

スループット係数TR

$$= \sum_{i=1}^{N} \frac{CPU_i + IO_i}{インタバル長} \quad \cdots\cdots\cdots (3)$$

上記の定義によれば、ジョブiの処理速度係数SR_iは、ジョブが単独で実行されている（経過時間の間、CPU処理か入出力処理のいずれかが行なわれている）時のジョブの処理速度を1とした場合の処理速度の比率（複数のジョブを同時に実行すると、ジョブ間の競合によりCPU処理待ち、入出力処理待ちが生じるので一般には1より小さくなる）を表わしていると考えられ、ジョブの平均処理速度係数ASRは各ジョブの処理速度係数SR_iの加重平均（経過時間$ETIME_i$による）になっている（但し、同一ジョブ内のCPU処理と入出力処理のオーバラップは無視できるものとしている）。一方、スループット係数TRは、ジョブが長時間待ち状態にならずに単独で実行されている（インタバルの間、CPU処理か入出力処理のいずれかが行なわれている）時のスループット（処理量）を1とした場合のスループットの比率（複数のジョブを同時に実行すると、ジョブ間のCPU処理、入出力処理のオーバラップが生じるので、一般には1より大きくなる）を表わしていると考えられる。

負荷計測機構50はジョブの実行を制御するジョブ実行制御機構46から次の事象が発生した場合に連絡を受けて負荷情報を算出するのに必要なデータを収集する。（第4図および第5図～第12図参照）

(1) ジョブ実行の開始／終了
(2) ジョブ実行の中断／再開
(3) 入出力処理の開始／終了
(4) 長時間待ち状態の発生／解除

ジョブの実行が開始されると、負荷情報を算出するのに必要なジョブごとの各種データを格納するジョブ・データ・テーブル51（第16図に詳細を示す）に当該ジョブ用のエントリを作成し、ジョブエントリ数をカウントアップ（＋1）した後、CPU時間計測中フラグ、経過時間計測中フラグをセットし、現在時刻をCPU時間計測開始時刻エリア、経過時間計測開始時刻エリアに格納する。（第5図）

ジョブの実行が中断されると、ジョブ・データ・テーブル51をサーチし当該ジョブ用のエントリを見つけ、現在時刻とCPU時間計測開始時刻の差をCPU時間に加えた後、CPU時間計測中フラグをリセットする。（第6図）ジョブの実行が再開されると、CPU時間計測中フラグをセットし、現在時刻をCPU時間計測開始時刻に格納する。（第7図）

入出力処理が開始されると、ジョブ・データ・テーブルをサーチし当該ジョブ用のエントリを見つけ、入出力実行中カウンタが0の場合は、現在時刻を入出力時間計測開始時刻に格納した後、入

出力実行中カウンタをカウントアップ（＋１）する。既に入出力時間の計測が開始されている（入出力実行中カウンタが０でない）場合は入出実行中カウンタのカウントアップ（＋１）のみ行なう。（第８図）入出力処理が終了すると、入出力実行中カウンタのカウント・ダウン（－１）を行ない、その結果入出力実行中カウンタが０になった場合は、現在時刻と入出力時間計測開始時刻の差を入出力時間に加える。まだ実行中の入出力がある場合は入出力時間の更新は行なわない。（第９図）

長時間待ち状態が発生／解除した場合は、ジョブ実行の中断／再開と同様にして経過時間の計測を中断／再開する。（第１０図、第１１図）

ジョブの実行が終了すると、ジョブ・データ・テーブルをサーチし当該ジョブ用のエントリを見つけ、ＣＰＵ時間、経過時間の更新を行なった後、ジョブ終了フラグをセットする。（第１２図）

ジョブ実行制御機構４６からの連絡を受けて収集されたこれらデータをもとに、負荷計測機構５０は計測インタバルごとに負荷情報を算出し、グローバル・プロセサの適正負荷決定機構４９へ送信する。（第４図および第１３図参照）

計測インタバルが経過すると、まずジョブ・データ・テーブルに登録されている各ジョブのＣＰＵ処理時間、入出力処理時間、経過時間を求める。ＣＰＵ処理時間は、ＣＰＵ時間計測中フラグがセットされていない場合はＣＰＵ時間の値を使用する。ＣＰＵ時間計測中フラグがセットされている場合はＣＰＵ時間の値に現在時刻とＣＰＵ時間計測開始時刻の差を加えてＣＰＵ処理時間を求めるとともに、現在時刻をＣＰＵ時間計測開始時刻に格納する。いずれの場合も、ＣＰＵ時間を０にしておく。入出力時間、経過時間についても同様にして求める。また、ジョブ終了フラグがセットされているジョブについては、当該エントリを削除し、ジョブ・エントリ数をカウント・ダウン(-1)しておく。各ジョブのＣＰＵ処理時間、入出力処理時間、経過時間が求まると、それらをもとにジョブの平均処理速度係数ＡＳＲ、スループット係数ＴＲを算出し、適正負荷決定機構４９へ送信する。ジョブ・データ・テーブルに登録されているジョブがなかった場合は、その旨適正負荷決定機構４９へ伝える。（第１３図）

適正負荷決定機構４９はグローバル・プロセサに設けられ、各プロセサの負荷計測機構５０から定期的に送られてくる負荷情報をもとに各プロセサにかかる負荷の適否を判断し、適正負荷を決定する。負荷を適正化するための手段として、本実施例ではバッチ・ジョブの処理多重度（アクティブなイニシエータの数）を制御する（同様にして、ＴＳＳジョブの処理多重度（アクティブなＴＳＳ端末の数）を制御することも可能である。（第１４図参照）

適正負荷決定機構４９は負荷情報を受け取ると、ジョブ・スケジュール制御テーブル４８（プロセサごとに、ジョブのスケジュールに関する情報を格納しているテーブルで、詳細を第１７図に示す）をサーチし当該プロセサ用のエントリを見つけ、送られてきたスループット係数ＡＳＲが予め定められた（ユーザが指定できる）スループット係数ＴＲの許容値をこえている場合はスループットが十分に出ており、現在の多重度が適正であると判断して現在の多重度（アクティブ・イニシエータ数）を適正多重度とする。スループット係数ＴＲが許容値に達していない場合は、ジョブの平均処理速度係数ＡＳＲを見て、平均処理速度係数ASRが予め定められた（ユーザ指定ができる）ジョブの平均処理速度係数ＡＳＲの許容範囲の下限値を下回っている場合は過剰に負荷がかかっていると見做して、現在の多重度から１を引いたものを適正多重度とする。平均処理速度係数ＡＳＲが許容範囲の上限値を上回っている場合は、現在の状態は負荷が不足していて、まだ負荷をかけてもよいと見做して、現在の多重度に１を加えたものを適正多重度とする。平均処理多重度係数が許容範囲に入っている場合は現在の多重度が適正であると判断して現在の多重度を適正多重度とする。（本実施例では多重度の増減値を１としているが、平均処理速度係数ＡＳＲの許容範囲からの外れ方に応じて増減値を変える方法も考えられる）。当該

計測インタバルにおいてジョブが1つも実行されなかった場合等で、ジョブの平均処理速度係数ASRが求まらなかった場合は、ジョブ・スケジュール制御テーブル48の総イニシエータ数の$\frac{1}{2}$を適正多重度とする。なお、算定された適正多重度は、予め定められた（ユーザ指定ができる）ジョブ・スケジュール制御テーブル48の多重度変更の範囲に入るように調整されて最終的な適正多重度が決定される。また、前の計測インタバルに比べて適正多重度を増やした場合は、ジョブが取り出せる可能性があるのでジョブ・スケジュール機構45の作動を指示する。

ジョブ・スケジュール機構45はジョブの実行をスケジュールするものであり、適正負荷決定機構49で適正多重度に従って実行待ちジョブの取り出しを制御することによりプロセサにかかる負荷の適正化を図る。（第15図参照）ジョブ・スケジュール機構45は、ジョブ入力機構44、適正負荷決定機構49から作動が指示された時、および、ジョブ実行制御機構46からジョブの取り出し要求、ジョブ実行の終了通知があった場合に作動する。ジョブ入力機構44はジョブの入力処理を行なうもので、カード・リーダ等からジョブ43を読込んでそれを後で実行可能なようにスプール・ボリューム5（第1図）へ格納し、ジョブ・キュー47へ登録した後、ジョブ・スケジュール機構45の作動を指示する。ジョブ・キュー47は入力された全ジョブについて、ジョブ名、ジョブ所有者名（ユーザ登録名）、ジョブ状態（ジョブが実行待ち状態にあるか、実行中か、出力待ち状態にあるか等を示す情報）等が格納されている。ジョブ名、ユーザ登録名はジョブ入力と同時にユーザによって入力される。

ジョブ・スケジュール機構45はジョブ入力機構、適正負荷決定機構49からの作動指示、ジョブ実行制御機構からのジョブの取り出し要求を受けると、ジョブ・スケジュール制御テーブルをサーチし、その時点でジョブの取り出し要求を出しているプロセサ用のエントリを見つけ、現在の多重度（アクティブイニシエータ数）が適正の多重度に達しているかどうかチェックし、既に適正多重度に到達している場合はジョブの取り出しを一時的に抑止する。適正多重度に到達していない場合はジョブ・キューから実行待ちジョブを取り出し、ジョブ実行制御機構にジョブを渡す。その後、アクティブ・イニシエータ数をカウントアップ（＋1）し、取り出されたジョブのジョブ状態を実行中にする。ジョブ実行制御機構46からジョブ実行の終了通知を受けた場合は、ジョブ・スケジュール制御テーブルをサーチし、当該プロセサ用のエントリを見つけ、アクティブ・イニシエータ数をカウント・ダウン（－1）し、実行が終了したジョブのジョブ状態を出力待ち状態にした後、上記のジョブ取り出し要求を受けた場合と同様の処理を行なう。

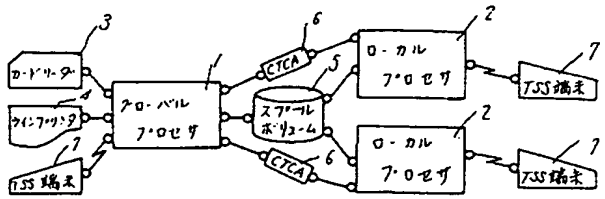
以上説明したごとく本発明によればLCMPシステムを構成する各プロセサにかかっている実負荷にもとづいてプロセサごとのジョブの適正多重度が決定され、それに従ってジョブの取り出しの制御がなされる。従って、オペレータの手を煩わすことなく、システム開始時の各プロセサの負荷状況に応じた適正なジョブ実行の多重度の設定ができるとともに、システム開始後の負荷変動（例えば、アクティブなTSS端末の増減）に応じたジョブ実行の多重度の動的な制御を行なうことができる。

図面の簡単な説明

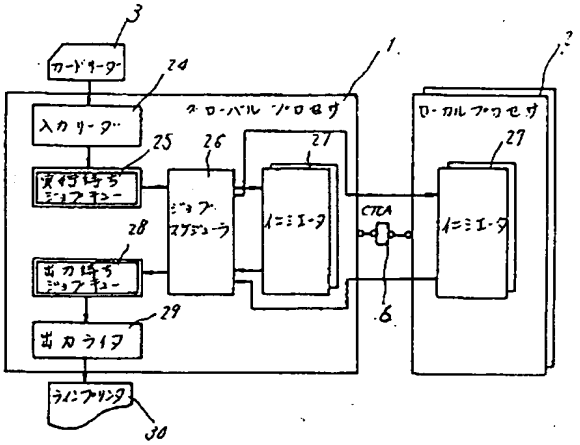
第1図はLCMPシステムの構成図を示す図、第2図は電子計算機システム（LCMP構成）を示す図、第3図は本発明の実施例を示す図、第4図～第14図は負荷計測処理のフロー・チャート、第15図はジョブ・スケジュール処理のフロー・チャート、第16図はジョブ・データ・テーブルの詳細を示す図、第17図はジョブ・スケジュール制御テーブルの詳細を示す図。

代理人　弁理士　薄　田　利　幸

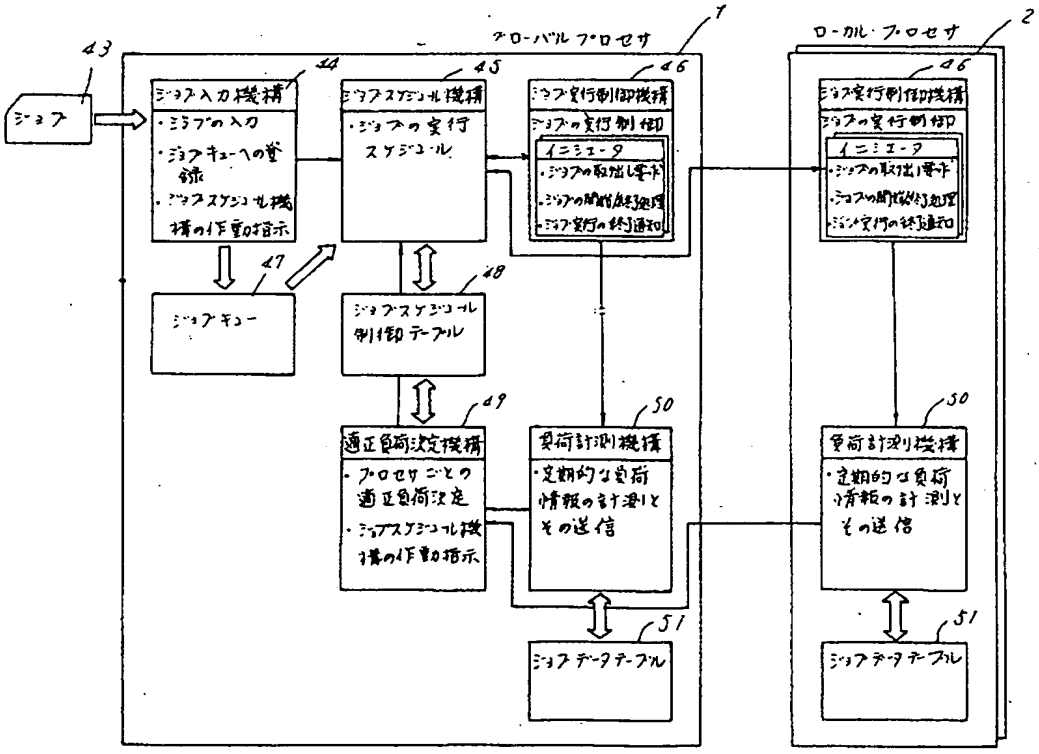
第 1 図



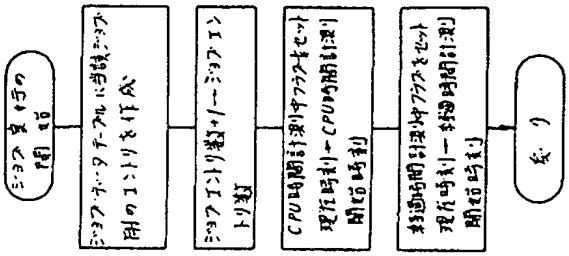
第 2 図



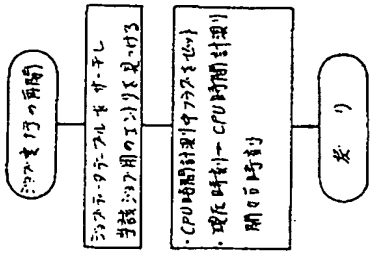
第 3 図



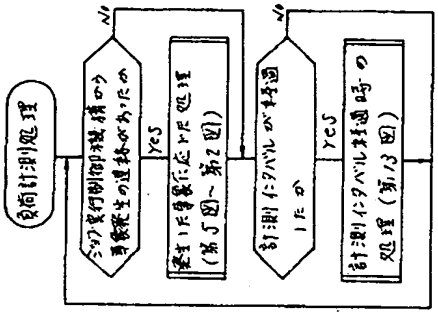
第 5 図



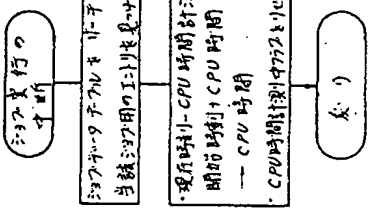
第 7 図



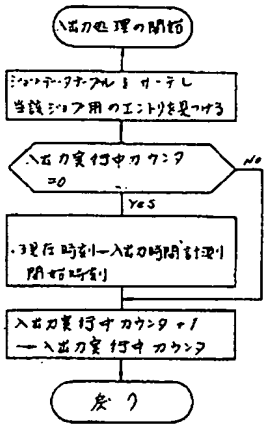
第 4 図



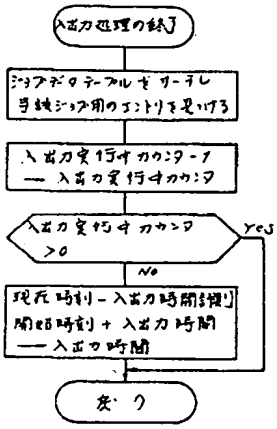
第 6 図



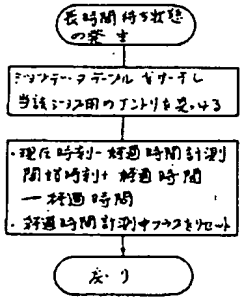
第 8 図



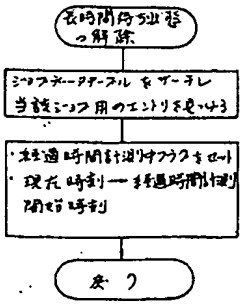
第 9 図



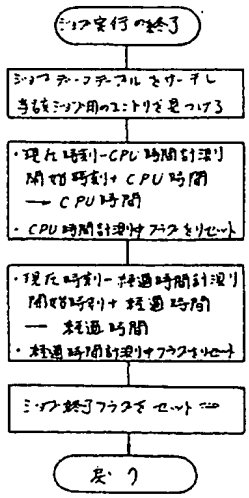
第 10 図



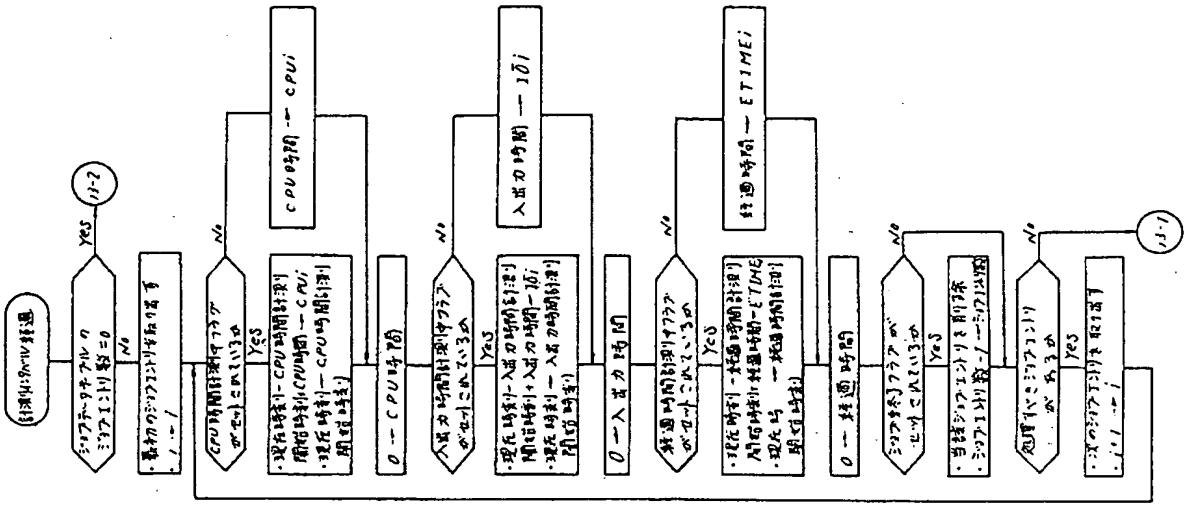
第 11 図



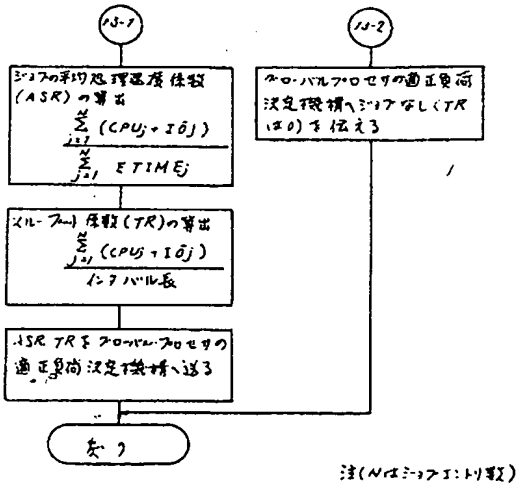
第 12 図



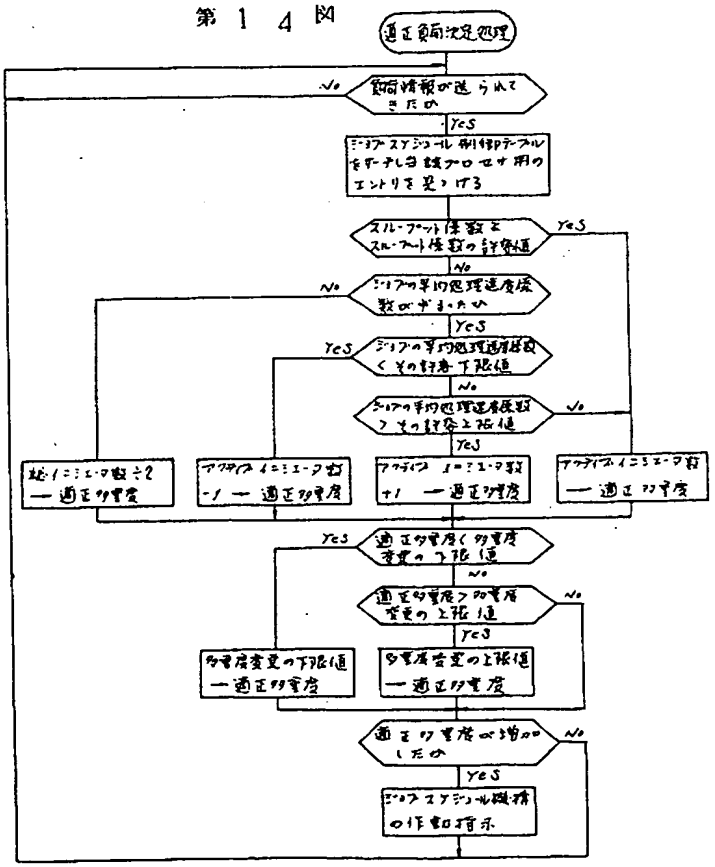
第１３図
（ａ）



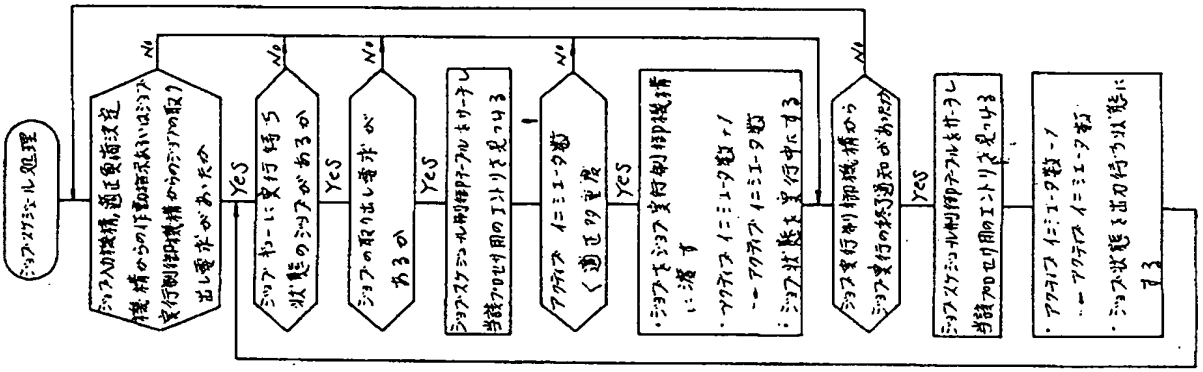
第１３図
（ｂ）



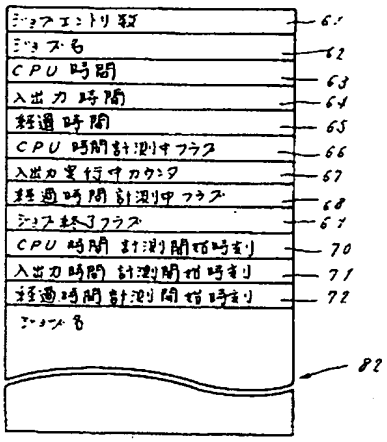
第１４図



第15図



第16図



第17図